1. Nick Wright CSC285 Problem 5 Linked List
2. Java Code

/\* This is a program that is capable of reading in student information from an input file, and putting it in a linked list of student objects.  
 \* The program currently contains the following three classes: myLinkManager, Student2, CodeforProb2CSC285 \*/  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
/\* class myLinkManager contains methods that can be used on a linked list  
\* it has methods getnumber, addnode, addfront, getnode, and addinorder\*/  
class myLinkManager<T extends Comparable>{  
 protected MyNode<T> head, tail; //head and tail of the list  
 protected int number;  
 public myLinkManager(){  
 MyNode<T> head = null; //set the head and the tail of the linked list to null  
 MyNode<T> tail = null;  
 int number = 0; //number is 0 because there is nothing in the list  
 }//end of constructor  
 public int getnumber(){  
 return number;  
 }  
 private static class MyNode<T>{  
 protected T nodevalue;  
 protected MyNode<T> next;  
 public MyNode(T x){  
 nodevalue=x;  
 next=null; //create pointer to the next node but set it to null  
 }//end of mynode constructor  
 }//end of class mynode  
  
 public int addnode(T x){  
 addfront(x);  
 return number;  
 }//end of addnode  
 public void addfront(T x){  
 //this will add a node to the front of the list  
 MyNode<T> newnode = new MyNode<T>(x);//creating a new node  
 //adding the node to the list  
 if(head==null){//this is if the list is empty  
 head=newnode;  
 tail = newnode;  
 }else{//this is if the list is not empty  
 //newnode will point to the current front of list  
 newnode.next=head;  
 //now the front of the list will point to new node  
 head = newnode;  
 }  
 //now we update the list by 1  
 number++;  
 return;  
 }//end of addfront  
 public T getnode(int x){  
 //this will return the node at the x'th position in the list  
 //if x is out of bounds an error message wil be printed and an exception will occur  
 if((x<0)||(x>number)){  
 System.*out*.println("error in getnode"+x+"while list holds"+number);  
 }  
 //iterator will start at the first node  
 int ict = 1;  
 MyNode<T> curnode;  
 //making the iterator point to the first node  
 curnode = head;  
 //setting up a loop that will iterate until curnode is equal to x  
 while(ict<x){  
 curnode = curnode.next;  
 ict++;  
 }  
 return curnode.nodevalue;  
 }//end of getnode  
 public void addinorder(T x){  
 //this will add nodes in order of their compareTo  
 //creating node with x information  
 MyNode<T> newnode = new MyNode<T>(x);  
 //two pointers being created, one pointing to the current position in the linked list and another pointing to the next position in the linked list  
 MyNode<T> cnode, nnode; //cnode = current node and nnode = nextnode  
 //checking for condition 1, condition 1 being that this is the first node in the list  
 if(number == 0){  
 //this will be the first node  
 head=newnode;  
 tail=newnode;  
 number=1;  
 return;  
 }  
 //next case is if there are already nodes in the list. We check to see if this goes in front, we are trying to sort nodes in descending order  
 if(x.compareTo(head.nodevalue)==1){  
 //this node goes in front  
 newnode.next=head;  
 head=newnode;  
 number++;  
 return;  
 }  
 if(x.compareTo(tail.nodevalue)==-1){  
 //this will put the node in the back  
 tail.next=newnode;  
 tail=newnode;  
 number++;  
 return;  
 }  
 //final case is if the node goes somewhere in between the front and back of the list  
 cnode=head;  
 nnode=head.next;  
 //now check and branch  
 while(x.compareTo(nnode.nodevalue)!=1){  
 cnode=nnode;//chain forward with current node  
 nnode=nnode.next;//chain forward with next node  
 }  
 //now x is greater than the nnode and less than cnode  
 //make the current node point to newnode and then make newnode point to the next node  
 cnode.next=newnode;  
 newnode.next=nnode;  
 number++;  
 return;  
 }//this is the end of addinorder  
}//this is the end of mylinkmanager  
  
/\* class Student2 contains the object Student2 constructors and variables used in Student2 objects  
 \* class Student2 implements Comparable so that the compareTo function may be overridden \*/  
class Student2 implements Comparable{  
 protected int pscore; //this is the percent score for the student  
 protected int[] testScore = new int[3]; //testScore is an integer array for the test scores of a student  
 protected String id; //String id is the variable to hold the student id  
 protected String name; //String name is the variable that will hold the ame of the student  
 protected String letterGrade; //a string variable that is used to store the letter grade  
 protected int TotalNoHours; //variable to hold the total hours a student has taken  
 protected float CumulativeGPA; //variable to hold a student's current GPA  
 protected String studentClass;//this variable holds the year of college that a student is is in  
 protected float newStudentGPA; //this variable will hold the updated GPA of a student based upon their GP in the current class  
  
 public Student2(){} //the default constructor for Student2  
 public Student2(int[] test, String id, String name, int TotalNoHours, float CumulativeGPA){ //Student2 object constructor with parameters  
 this.id=id;  
 this.name=name;  
 for(int i=0; i<3; i++){ //for loop is used to put the test scores in the testScore array  
 this.testScore[i]=test[i];  
 }  
 this.TotalNoHours=TotalNoHours;  
 this.CumulativeGPA=CumulativeGPA;  
  
 pscore = (int)(((test[0]+test[1]+test[2])/3.0)+0.5); //calculates the average test score as an integer  
 if(pscore >= 90){ //if else statement chain that will calculate letterGrade based on the pscore variable  
 letterGrade = "A";  
 }else if((pscore<90) && (pscore>=80)){  
 letterGrade = "B";  
 }else if((pscore<80) && (pscore>=70)){  
 letterGrade = "C";  
 }else if((pscore<70) && (pscore>=60)){  
 letterGrade = "D";  
 }else{  
 letterGrade = "F";  
 }//end of if else chain  
  
 if(TotalNoHours <= 30){ //if else statement chain is used to calculate which grade a student is in depending on the amount of credit hours they have taken.  
 studentClass = "FR";  
 }else if((TotalNoHours > 30) && (TotalNoHours <= 60)){  
 studentClass = "SO";  
 }else if((TotalNoHours > 60)&&(TotalNoHours<=90)){  
 studentClass = "JR";  
 }else if(TotalNoHours>90){  
 studentClass = "SR";  
 } //end of if else chain  
  
  
 if(letterGrade.equals("A")){ //if else chain will calculate the new GPA of a student, variations in the formula occur depending on the student's letter grade in the class.  
 newStudentGPA = ((CumulativeGPA\*TotalNoHours)+(2)\*4)/(TotalNoHours+2); //a letter grade of A means that we multiply by 4  
 newStudentGPA = (int)(newStudentGPA\*100); //I am getting rid of extra decimals in the GPA, so that they have a maximum of 2 decimal places  
 newStudentGPA = newStudentGPA/100;  
 }else if(letterGrade.equals("B")){  
 newStudentGPA = ((CumulativeGPA\*TotalNoHours)+(2)\*3)/(TotalNoHours+2); //a letter grade of B means that we multiply by 3  
 newStudentGPA = (int)(newStudentGPA\*100);  
 newStudentGPA = newStudentGPA/100;  
 }else if(letterGrade.equals("C")){  
 newStudentGPA = ((CumulativeGPA\*TotalNoHours)+(2)\*2)/(TotalNoHours+2); //a letter grade of C means that we multiply by 2  
 newStudentGPA = (int)(newStudentGPA\*100);  
 newStudentGPA = newStudentGPA/100;  
 }else if(letterGrade.equals("D")){  
 newStudentGPA = ((CumulativeGPA\*TotalNoHours)+(2)\*1)/(TotalNoHours+2); //a letter grade of D means that we multiply by 1  
 newStudentGPA = (int)(newStudentGPA\*100);  
 newStudentGPA = newStudentGPA/100;  
 }else{  
 newStudentGPA = ((CumulativeGPA\*TotalNoHours)+(2)\*0)/(TotalNoHours+2); //a letter grade of F means that we multiply by 0  
 newStudentGPA = (int)(newStudentGPA\*100);  
 newStudentGPA = newStudentGPA/100;  
 }//end of if else chain  
 }//end of Student2 object constructor  
  
 public int getPscore() { //a getter method for pscore that is used in the comapreTo method  
 return pscore;  
 }// a getter method for returning the pscore variable  
  
 @Override  
 public String toString() { //changing the format for when we want to print out an object  
 return id+" | "+name+" | "+testScore[0]+" | "+testScore[1]+" | "+testScore[2]+" | "+pscore+"% | "+  
 letterGrade+" | "+TotalNoHours+" | "+CumulativeGPA+" | "+newStudentGPA+" | "+studentClass;  
 }  
  
 @Override  
 public int compareTo(Object o) { //changing compareTo function for when we use it in method SortLarge  
 if(getPscore()>((Student2)o).getPscore()){  
 return 1; //returning 1 if object 1 is larger than object 2  
 } else if (getPscore()<((Student2)o).getPscore()){  
 return -1; //returning -1 if object 1 is smaller than object 2  
 } else {  
 return 0; //will return 0 if objects are the same  
 }  
 }  
}//end of student2 class  
  
/\* public class CodeforProb2CSC285 contains the main method\*/  
public class CodeforProb2CSC285{  
  
 public static void main(String[] args) throws FileNotFoundException { //FileNotFoundException needed for reading the input file  
 int i, num; //creating integer variables i and num  
 myLinkManager<Student2> linkstu = new myLinkManager<Student2>(); //creating a Linked List for student2 objects  
  
 Scanner input = new Scanner(new File("Input2.txt"));  
 while(input.hasNext()){ //while loop used to iterate through the input file and assign variables their values  
 //creating variables that will be used to temporarily store the values being read in from input text before they are put into an object  
 int[] test = new int[3];  
 String Sid = input.next();  
 String name = input.next();  
 test[0] = input.nextInt();  
 test[1] = input.nextInt();  
 test[2] = input.nextInt();  
 int creditHour = input.nextInt();  
 float gpa = input.nextFloat();  
  
 Student2 workStu2 = new Student2(test, Sid, name, creditHour, gpa); //creating a new Student2 object  
 linkstu.addinorder(workStu2); //storing object in linkstu  
 }//end of while loop  
  
 System.*out*.println("\n" + "----------------------------------------------------------------------------------------------" + "\n" +  
 "Part 2 Student Objects using a Linked List"  
 + "\n" + "----------------------------------------------------------------------------------------------");  
 System.*out*.println("List the objects from the Liked List including the % score and the grades.");  
 System.*out*.println("Student ID | Name | Test 1 | Test 2 | Test 3 | Percent Score | Letter Grade | Credit Hours | GPA before entering class | Updated GPA | Year of Schooling");  
  
 //iterating through the linked list and printing out the objects  
 num = linkstu.getnumber();  
 for(i=1;i<=num;i++){  
 System.*out*.println(linkstu.getnode(i));  
 }  
  
 //adding new students to the array list  
 linkstu.addinorder(new Student2(new int[]{80,75,98},"67T4","Clouse,B", 102, (float)3.65));  
 linkstu.addinorder(new Student2(new int[]{75,78,72},"45P5","Garrison,J", 39, (float)1.85));  
 linkstu.addinorder(new Student2(new int[]{85,95,99},"89P0","Singer,A", 130, (float)3.87));  
  
 System.*out*.println("\n" + "List the linked List after the new students have been added and their grades have been sorted from highest to lowest.");  
 System.*out*.println("Student ID | Name | Test 1 | Test 2 | Test 3 | Percent Score | Letter Grade | Credit Hours | GPA before entering class | Updated GPA | Year of Schooling");  
  
 //iterating through the linked list and printing out the objects  
 num = linkstu.getnumber();  
 for(i=1;i<=num;i++){  
 System.*out*.println(linkstu.getnode(i));  
 }  
 } //end of main method  
} //end of CodeforProb2CSC285

1. Input Files

45A3 Jones,H 86 88 95 98 3.42

34K5 Horner,M 67 75 23 17 1.95

56J8 Gach,T 75 85 90 60 3.75

34U8 Hunter,C 100 50 75 75 2.60

42P4 Hinrichs,S 85 75 65 52 3.29

78T6 Johnson,K 80 78 89 15 2.00

44L2 Levitte,H 56 66 99 100 2.35

88I9 Garner,J 95 92 98 110 3.89

1. Output Files

----------------------------------------------------------------------------------------------

Part 2 Student Objects using a Linked List

----------------------------------------------------------------------------------------------

List the objects from the Liked List including the % score and the grades.

Student ID | Name | Test 1 | Test 2 | Test 3 | Percent Score | Letter Grade | Credit Hours | GPA before entering class | Updated GPA | Year of Schooling

88I9 | Garner,J | 95 | 92 | 98 | 95% | A | 110 | 3.89 | 3.89 | SR

45A3 | Jones,H | 86 | 88 | 95 | 90% | A | 98 | 3.42 | 3.43 | SR

56J8 | Gach,T | 75 | 85 | 90 | 83% | B | 60 | 3.75 | 3.72 | SO

78T6 | Johnson,K | 80 | 78 | 89 | 82% | B | 15 | 2.0 | 2.11 | FR

34U8 | Hunter,C | 100 | 50 | 75 | 75% | C | 75 | 2.6 | 2.58 | JR

42P4 | Hinrichs,S | 85 | 75 | 65 | 75% | C | 52 | 3.29 | 3.24 | SO

44L2 | Levitte,H | 56 | 66 | 99 | 74% | C | 100 | 2.35 | 2.34 | SR

34K5 | Horner,M | 67 | 75 | 23 | 55% | F | 17 | 1.95 | 1.74 | FR

List the linked List after the new students have been added and their grades have been sorted from highest to lowest.

Student ID | Name | Test 1 | Test 2 | Test 3 | Percent Score | Letter Grade | Credit Hours | GPA before entering class | Updated GPA | Year of Schooling

88I9 | Garner,J | 95 | 92 | 98 | 95% | A | 110 | 3.89 | 3.89 | SR

89P0 | Singer,A | 85 | 95 | 99 | 93% | A | 130 | 3.87 | 3.87 | SR

45A3 | Jones,H | 86 | 88 | 95 | 90% | A | 98 | 3.42 | 3.43 | SR

67T4 | Clouse,B | 80 | 75 | 98 | 84% | B | 102 | 3.65 | 3.63 | SR

56J8 | Gach,T | 75 | 85 | 90 | 83% | B | 60 | 3.75 | 3.72 | SO

78T6 | Johnson,K | 80 | 78 | 89 | 82% | B | 15 | 2.0 | 2.11 | FR

34U8 | Hunter,C | 100 | 50 | 75 | 75% | C | 75 | 2.6 | 2.58 | JR

42P4 | Hinrichs,S | 85 | 75 | 65 | 75% | C | 52 | 3.29 | 3.24 | SO

45P5 | Garrison,J | 75 | 78 | 72 | 75% | C | 39 | 1.85 | 1.85 | SO

44L2 | Levitte,H | 56 | 66 | 99 | 74% | C | 100 | 2.35 | 2.34 | SR

34K5 | Horner,M | 67 | 75 | 23 | 55% | F | 17 | 1.95 | 1.74 | FR

Process finished with exit code 0

1. Class Documentation

**class myLinkManager**<T extends Comparable>

**myLinkManager Data**

protected MyNode<T> head, tail;

protected int number;

**myLinkManager Functions**

private static class MyNode<T>

public myLinkManager()

public int addnode(T x)

public void addfront(T x)

public T getnode(int x)

public void addinorder(T x)

**class Student2** implements Comparable

**Student2 Data**

protected int pscore;

protected int[] testScore = new int[3];

protected String id;

protected String name;

protected String letterGrade;

protected int TotalNoHours;

protected float CumulativeGPA;

protected String studentClass;

protected float newStudentGPA;

**Student2 Functions**

public Student2()

public Student2(int[] test, String id, String name, int TotalNoHours, float CumulativeGPA)

public int getPscore()

public String toString()

public int compareTo(Object o)

public **class CodeforProb2CSC285**

**CodeforProb2CSC285 Data**

The following Data is within the main method of this class:

int i, num;

myLinkManager<Student2> linkstu = new myLinkManager<Student2>();

Scanner input = new Scanner(new File("Input2.txt"));

**CodeforProb2CSC285 Functions**

public static void main(String[] args) throws FileNotFoundException